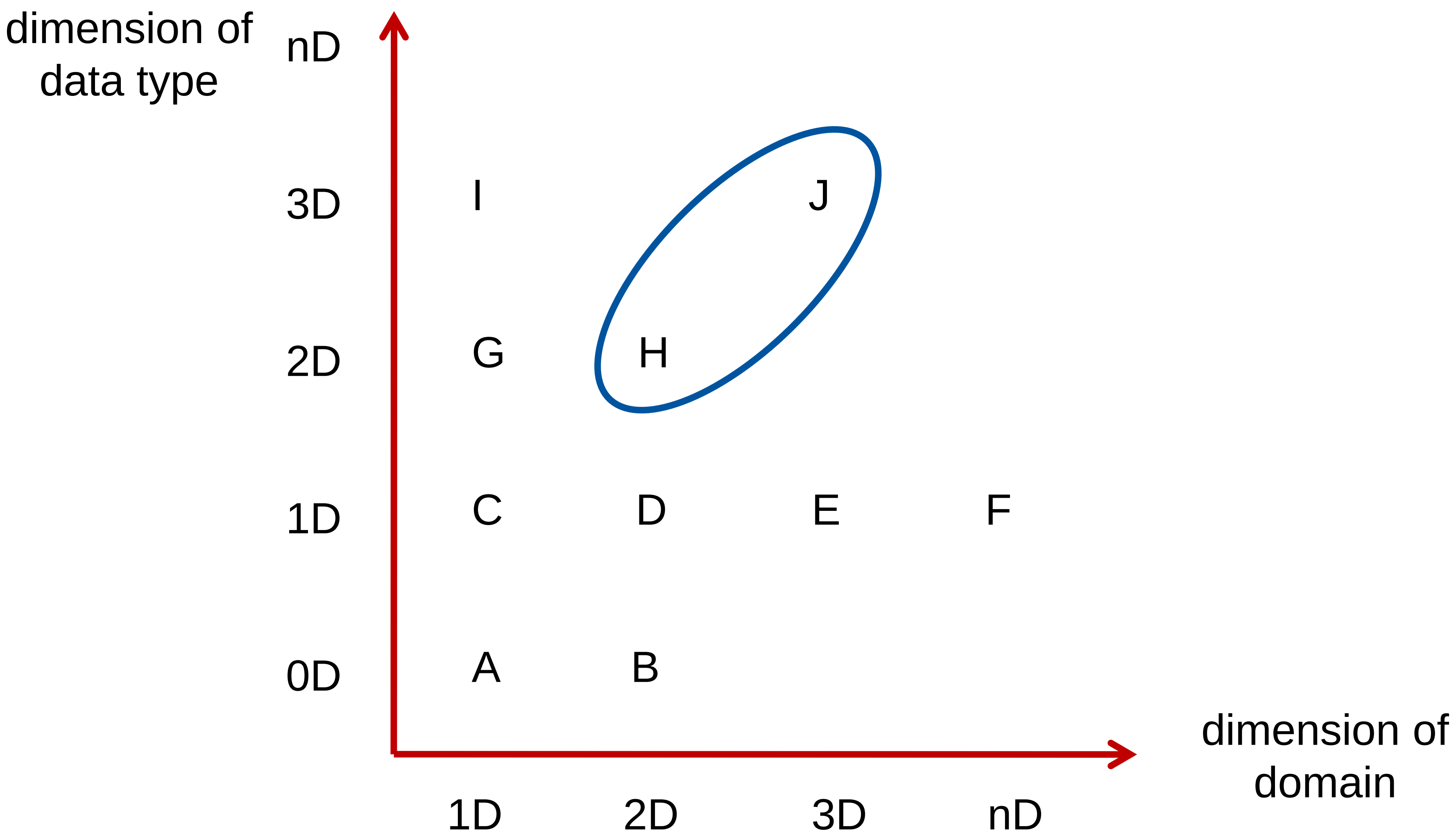
Vector Field Visualization

Leif Kobbelt

Visual Computing Institute | RWTH AACHEN UNIVERSITY

Types of Data



Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Characteristic Lines

- Types of characteristic lines in a vector field:
  - stream lines: tangential to the vector field
  - path lines: trajectories of massless particles in the flow (non-static flow fields)
  - streak lines: trace of dye that is deposited into the flow at a fixed position
  - time lines (time surfaces): propagation of a line (surface) of massless elements in time

Characteristic Lines

- stream lines
  - tangential to the vector field
  - stationary vector field (arbitrary, yet fixed time *t*)
  - stream line is a solution to the initial value problem of an ordinary differential equation:

$$\mathbf{L}(0) = \mathbf{x}_0 \quad , \quad \frac{d\mathbf{L}(u)}{du} = \mathbf{v}(\mathbf{L}(u))$$

initial value
(seed point x_0)

ordinary differential equation

Characteristic Lines

- path lines
  - trajectories of massless particles in the flow
  - vector field can be time-dependent (unsteady)
  - path line is a solution to the initial value problem of an ordinary differential equation:

$$\mathbf{L}(0) = \mathbf{x}_0 \quad , \quad \frac{d\mathbf{L}(t)}{dt} = \mathbf{v}(\mathbf{L}(t), t)$$

Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Characteristic Lines

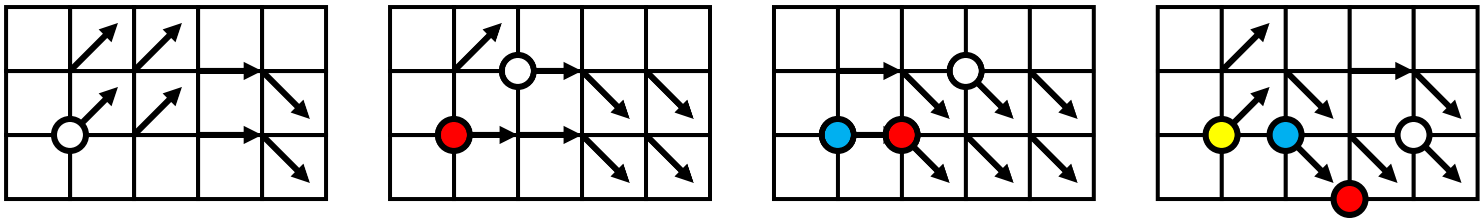
- streak lines
  - trace of dye that is released into the flow at a fixed position
  - connect all particles that passed through a certain position (non-stationary flow)

- time lines (time surfaces)
  - propagation of a line (surface) of massless elements over time
  - many point-like particles that are traced synchronously
  - connect particles that were deposited simultaneously

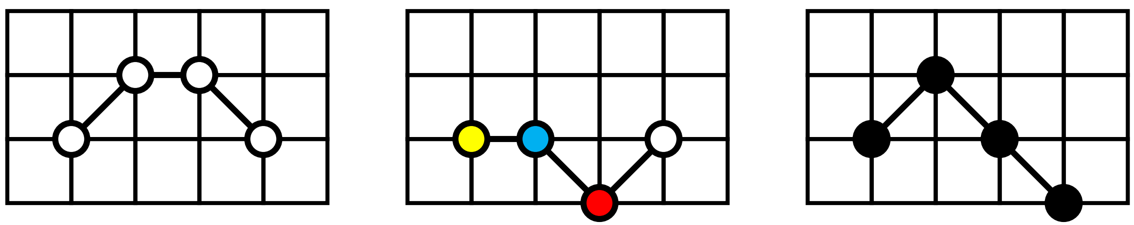
Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Characteristic Lines

- comparison of path lines, streak lines, and stream lines



t_0 t_1 t_2 t_3

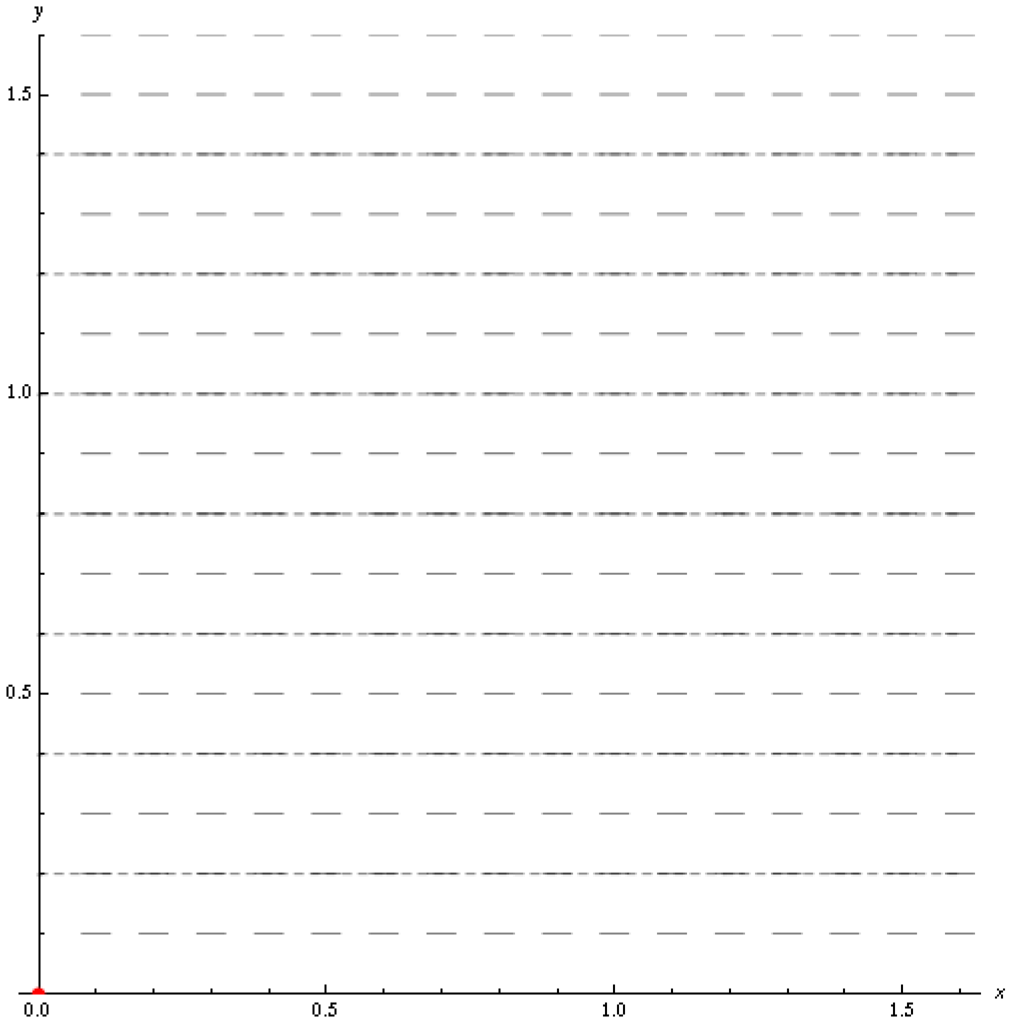


path line streak line stream line for t_3

- path lines, streak lines, and stream lines are identical for stationary flows

Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Characteristic Lines



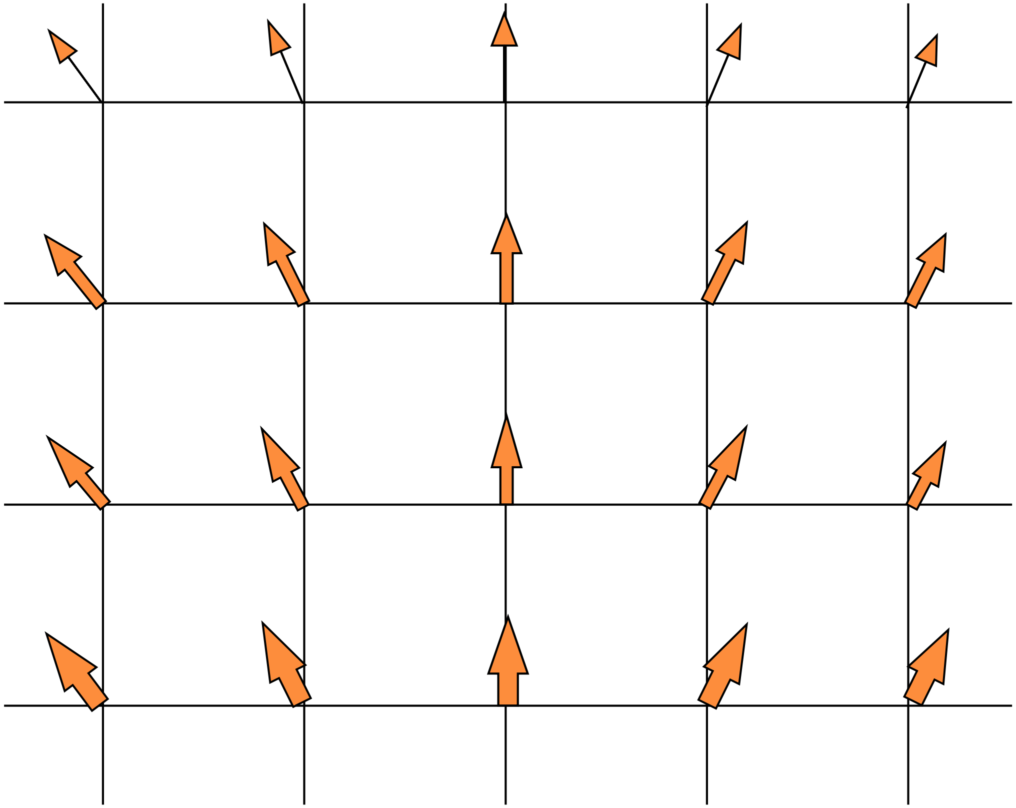
Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Arrows and Glyphs

- visualize **local** features of the vector field:
  - vector itself
  - vorticity
  - external data: temperature, pressure, etc.
- important elements of a vector:
  - direction
  - magnitude
  - not: components of a vector
- approaches:
  - arrow plots
  - glyphs

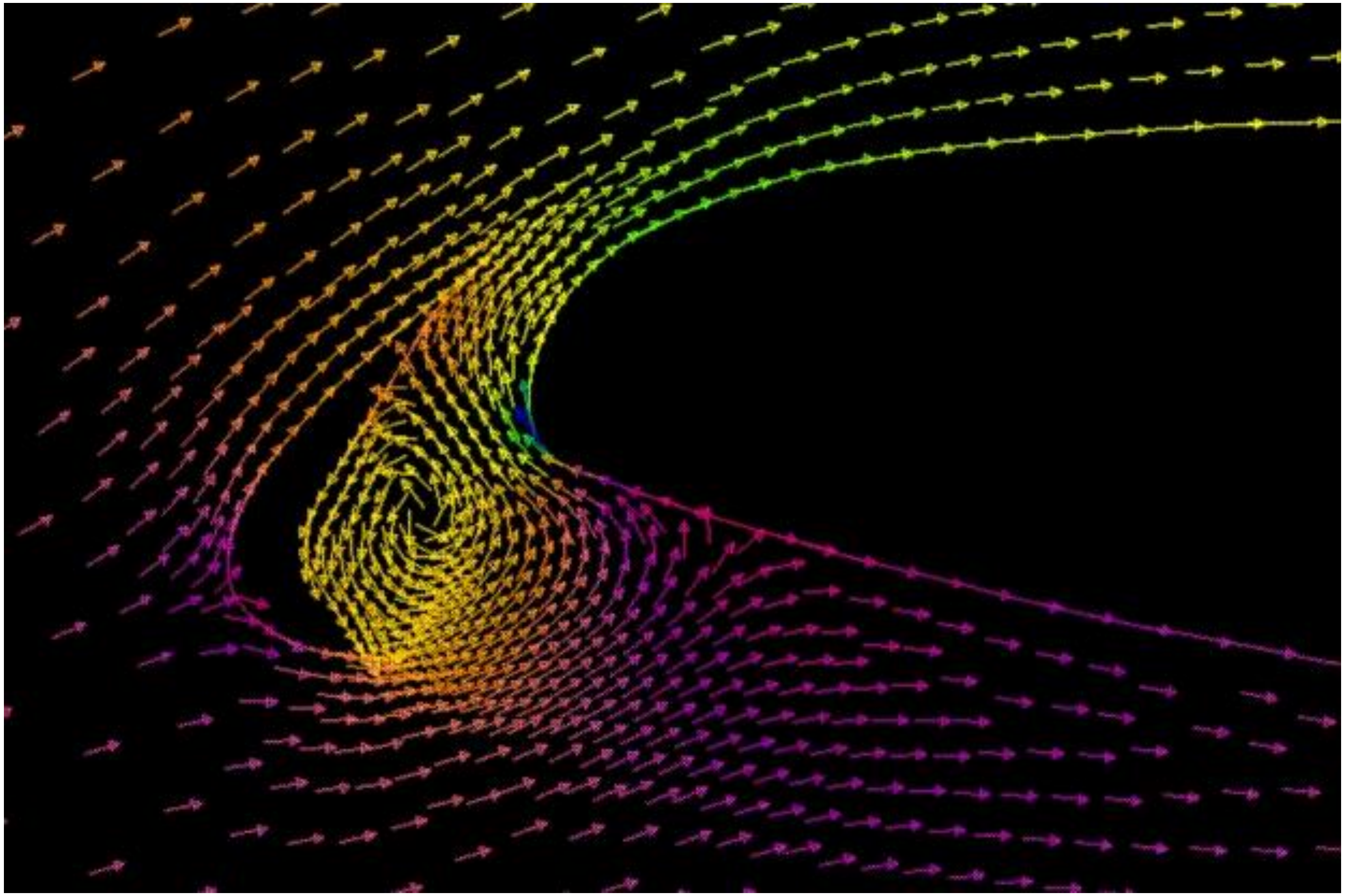
Arrows and Glyphs

- arrows visualization
  - direction of vector field
  - orientation
  - magnitude:
    - length of arrows
    - color coding



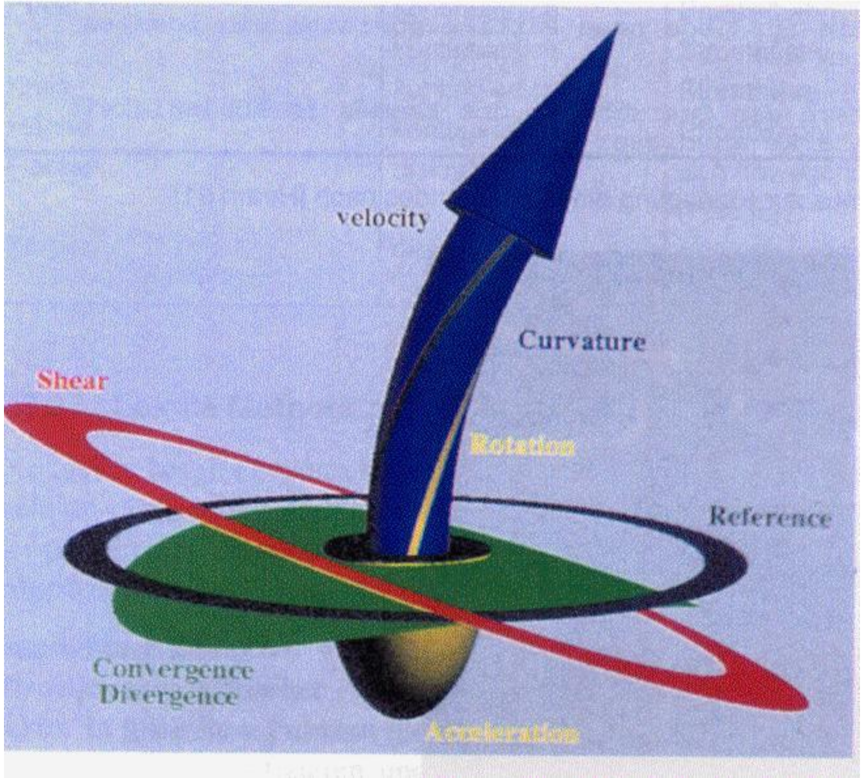
Arrows and Glyphs

- arrows



Arrows and Glyphs

- glyphs
  - can visualize more features of the vector field (flow field)



Arrows and Glyphs

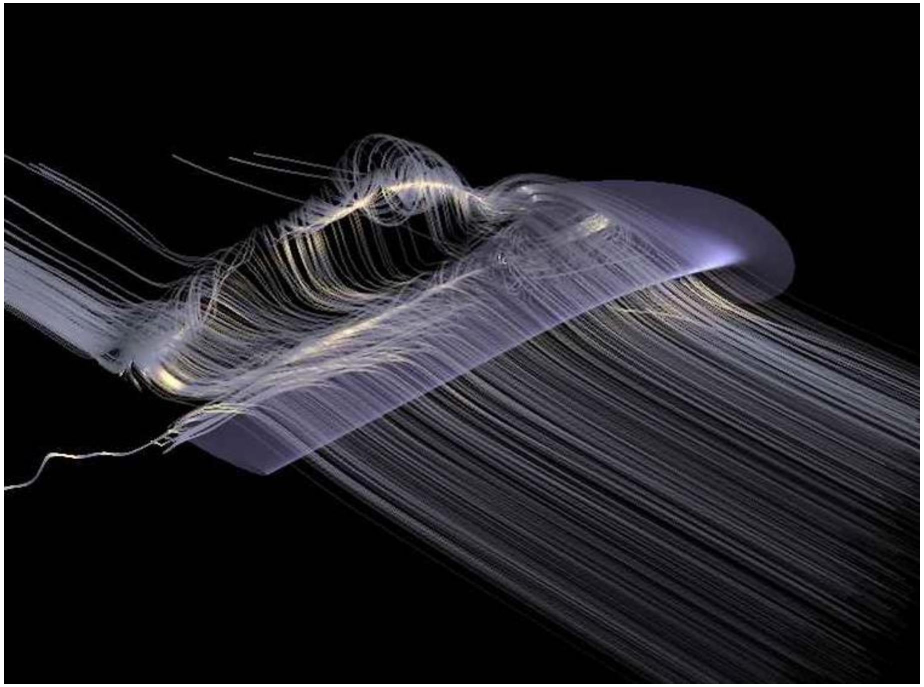
- pros and cons of glyphs and arrows:
  + simple
  + 3D effects
  - heavy load in the graphics subsystem
  - inherent occlusion effects
  - poor results if magnitude of velocity changes rapidly (use arrows of constant length and color code magnitude)

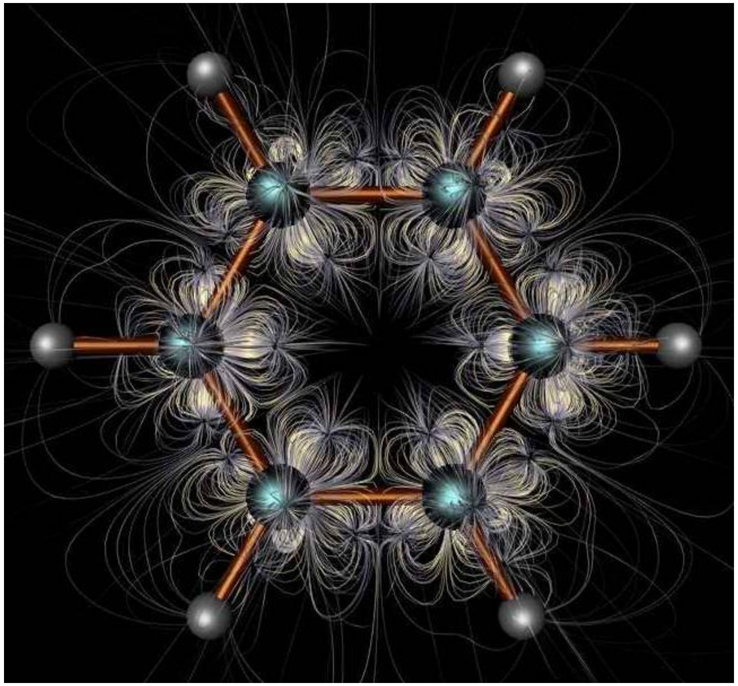
Mapping Methods Based on Particle Tracing

- basic idea: trace particles
- characteristic lines
- **local** or **global** methods
- mapping approaches:
  - lines
  - surfaces
  - individual particles
  - sometimes animated

Mapping Methods Based on Particle Tracing

- path lines

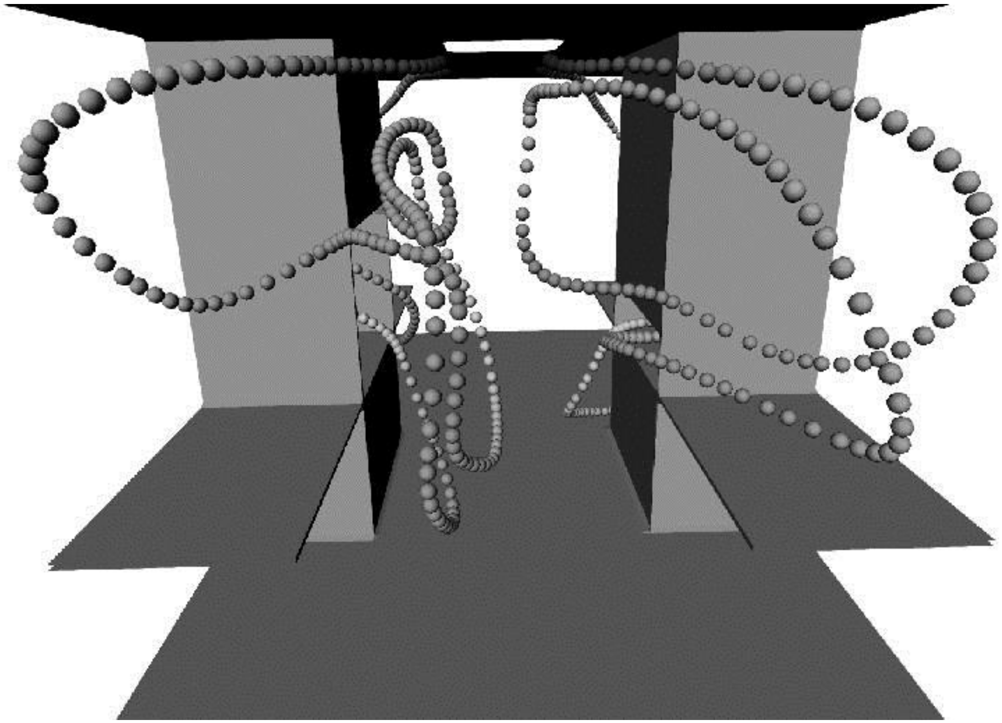




Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

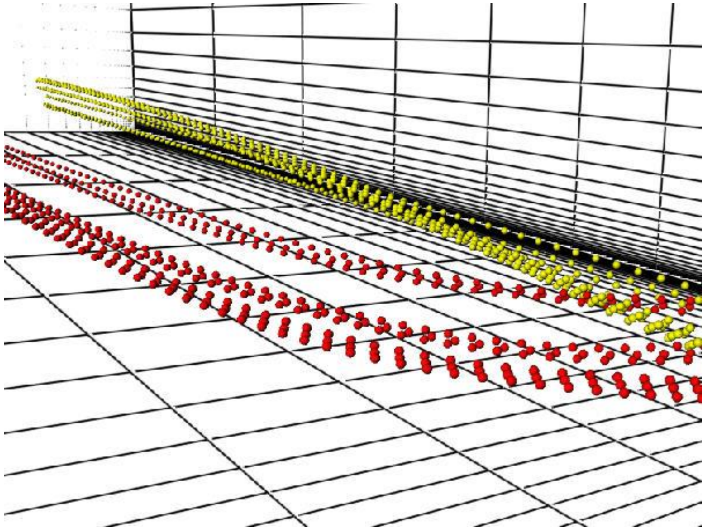
Mapping Methods Based on Particle Tracing

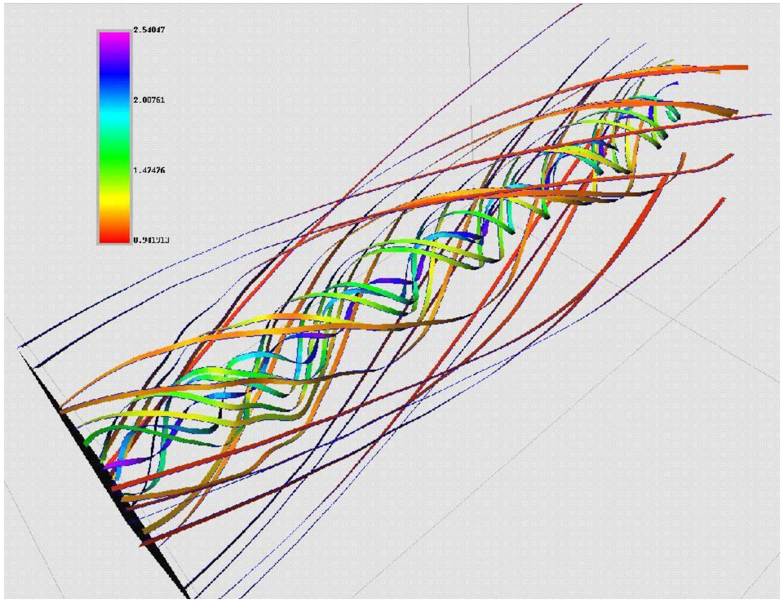
- stream balls
  - encode additional scalar value by radius

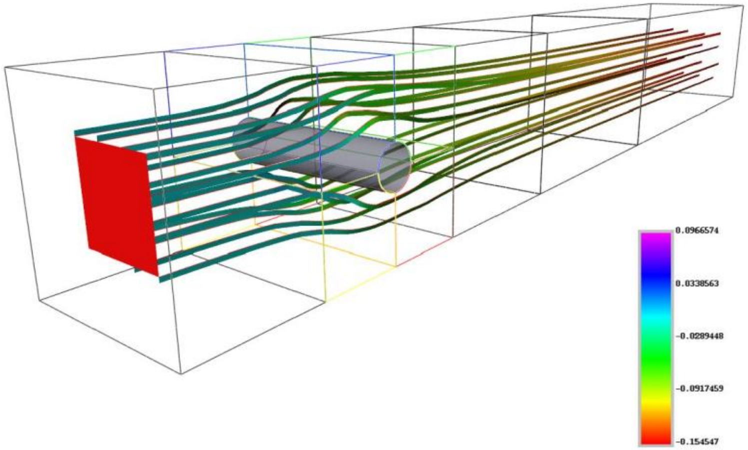


Mapping Methods Based on Particle Tracing

- streak lines



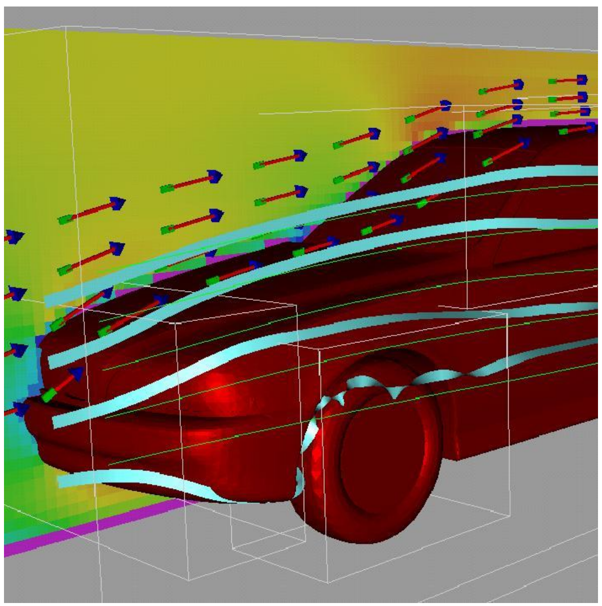


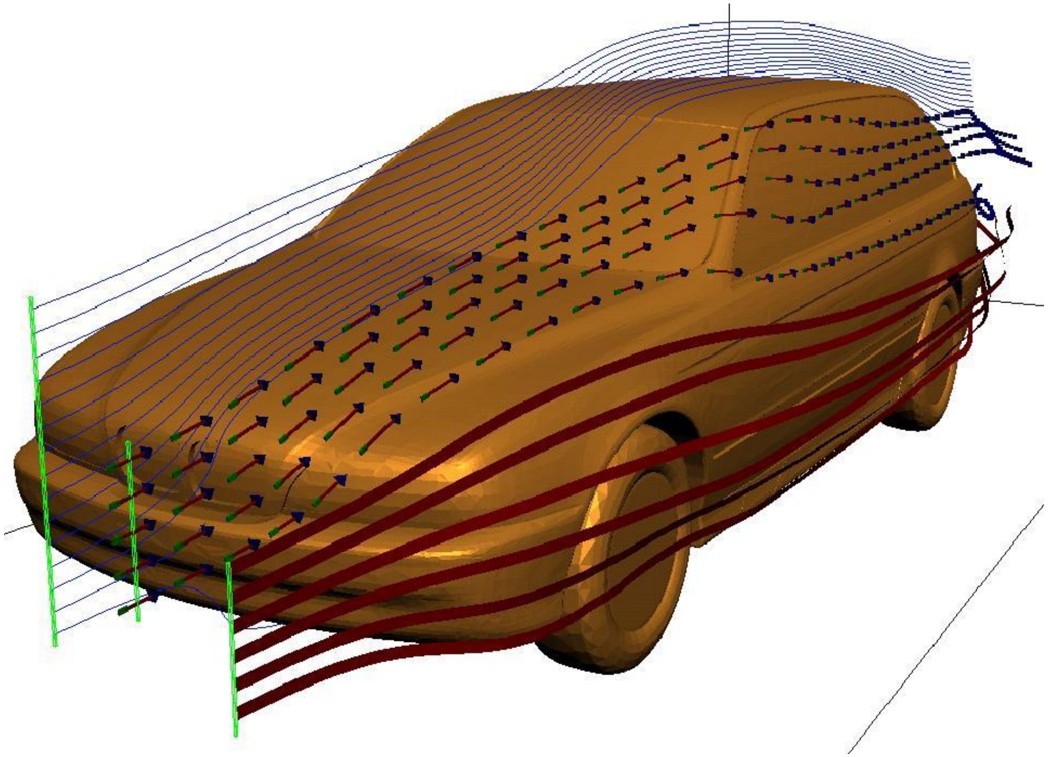


Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Mapping Methods Based on Particle Tracing

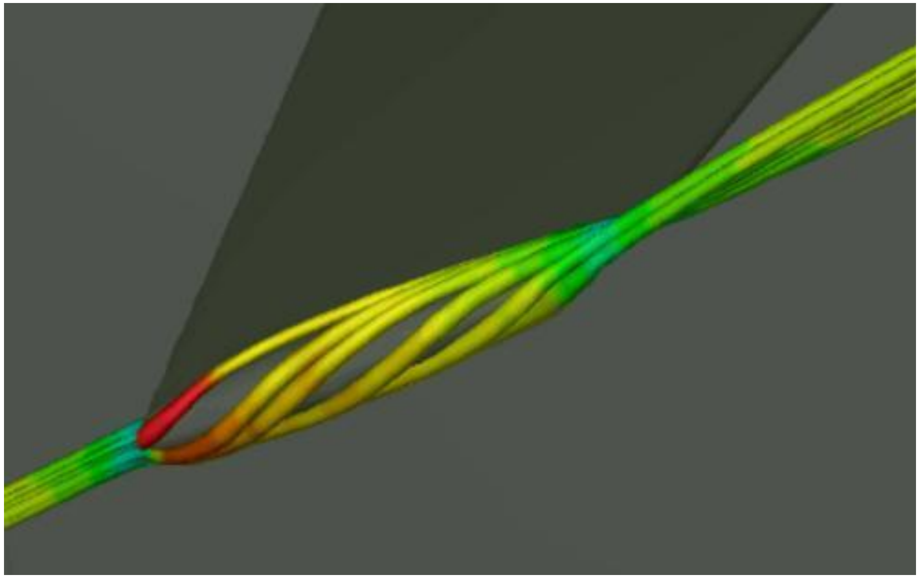
- stream ribbons
  - trace two close-by particles
  - keep distance constant

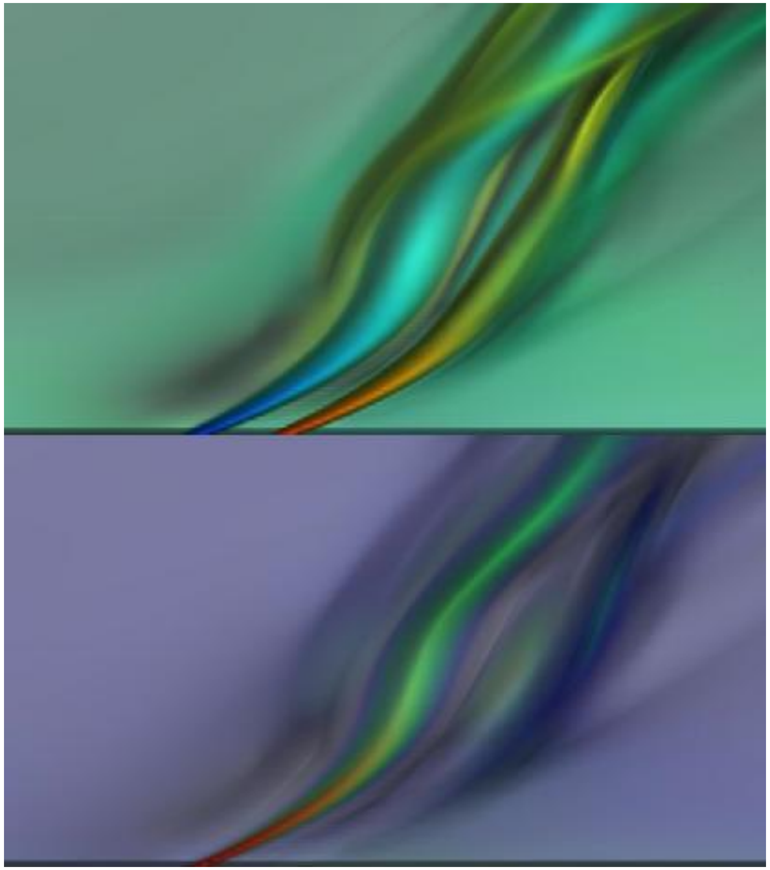




Mapping Methods Based on Particle Tracing

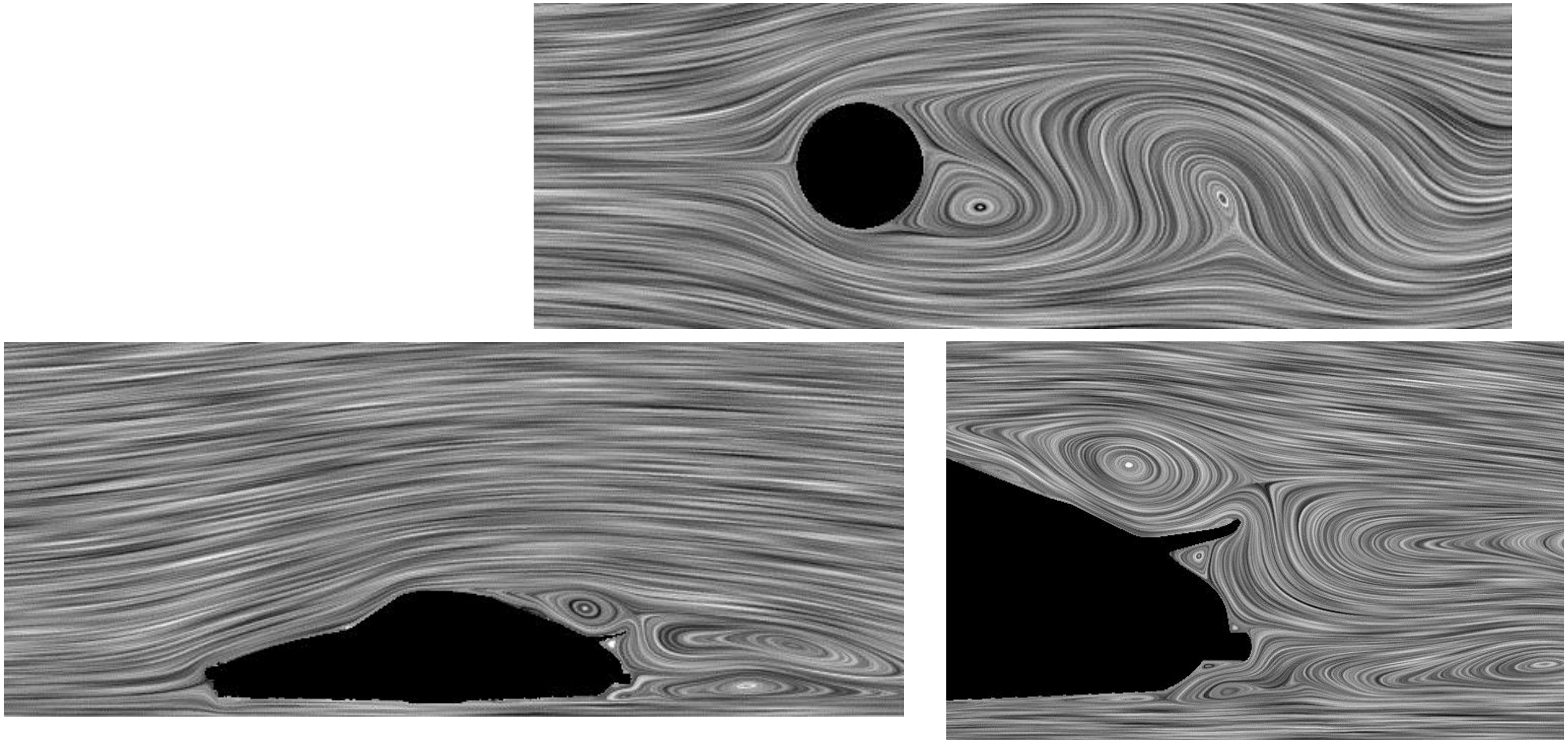
- stream tubes
  - specify contour, e.g. triangle or circle, and trace it through the flow





Mapping Methods Based on Particle Tracing

- LIC (Line Integral Convolution)



Numerical Integration of ODEs

- typical example of particle tracing problem (path line):

$$\mathbf{L}(0) = \mathbf{x}_0 \quad , \quad \frac{d\mathbf{L}(t)}{dt} = \mathbf{v}(\mathbf{L}(t), t)$$

- initial value problem for ordinary differential equations (ODE)
- what kind of numerical solver?

Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Numerical Integration of ODEs

- rewrite ODE in generic form
- initial value problem for:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, t)$$

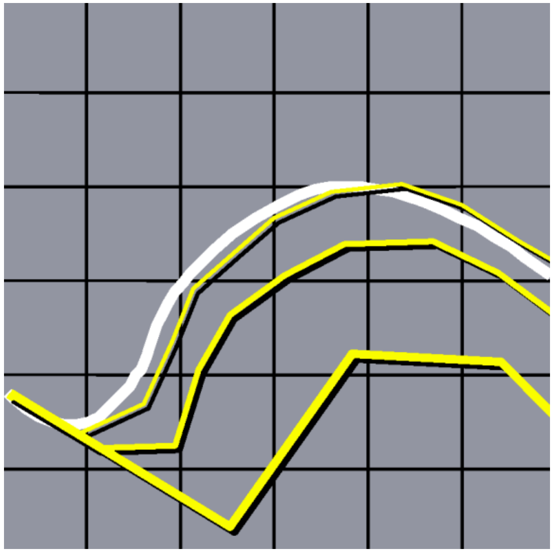
- most simple (naive) approach: explicit Euler method

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{f}(\mathbf{x}, t)$$

- based on Taylor expansion

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \dot{\mathbf{x}}(t) + O(\Delta t^2)$$

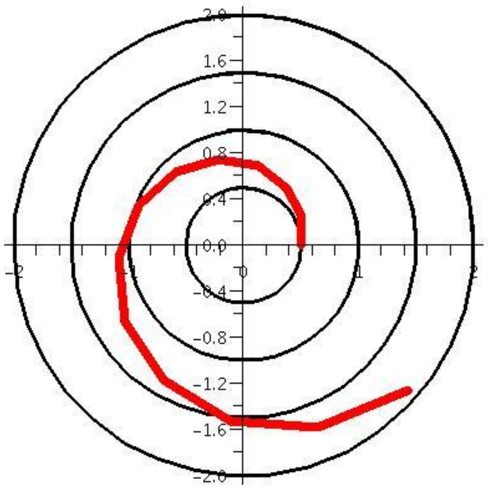
- first order method
- increase accuracy by smaller step size



Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Numerical Integration of ODEs

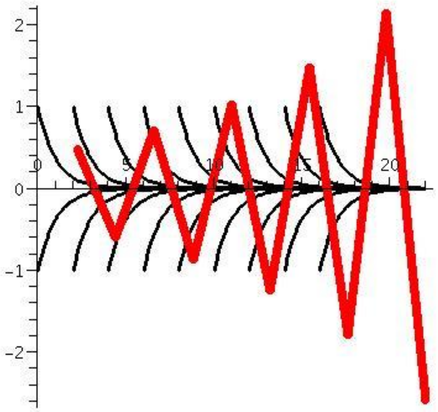
- Problem of explicit Euler method
  – inaccurate
  – unstable



- Example:

$$\dot{x} = -kx$$

$$x = e^{-kt}$$



divergence for $\Delta t > 2/k$

Numerical Integration of ODEs

- Implicit Euler method
  - Approximation of derivative

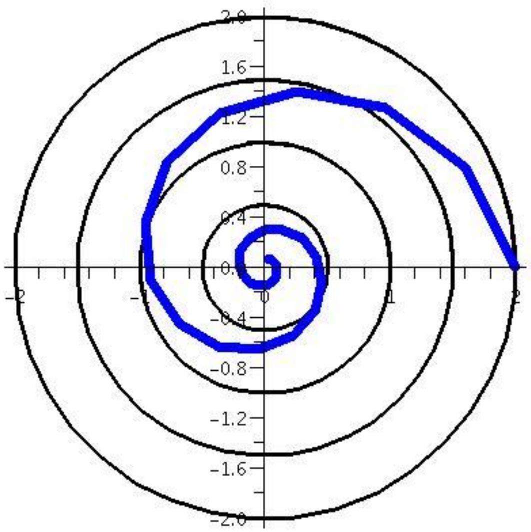
$$\dot{\mathbf{x}}(t+\Delta t) \approx \frac{x(t+\Delta t)-x(t)}{\Delta t}$$

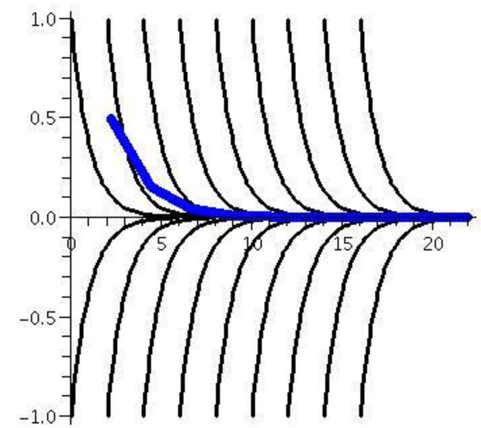
  - Implicit timestep

$$x(t+\Delta t) = x(t) + \Delta t \cdot \dot{\mathbf{x}}(t+\Delta t)$$

  - Solution of linear system due to the deriv new position

- Also method of first order





Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Visual Computing Institute
RWTH AACHEN UNIVERSITY

Numerical Integration of ODEs

- Midpoint method

  1. explicit Euler step

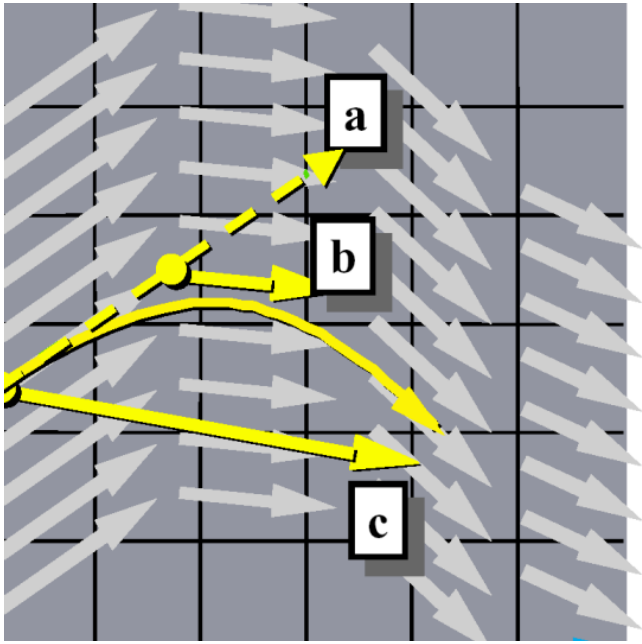
  $$\Delta \mathbf{x} = \Delta t \, \mathbf{f}(\mathbf{x}, t)$$

  2. Evaluation of **f** at midpoint

  $$\mathbf{f}_{\text{mid}} = \mathbf{f}\left(\mathbf{x} + \frac{\Delta \mathbf{x}}{2}, t + \frac{\Delta t}{2}\right)$$

  3. Complete step with value at midpoint

  $$\mathbf{x}(t + \Delta t) = x(t) + \Delta t \, \mathbf{f}_{\text{mid}}$$



- Method of second order

Numerical Integration of ODEs

- Classical Runge-Kutta of fourth order

$$\mathbf{k}_1 = \Delta t\, \mathbf{f}(\mathbf{x}, t)$$

$$\mathbf{k}_2 = \Delta t\, \mathbf{f}\left(\mathbf{x} + \frac{\mathbf{k}_1}{2}, t + \frac{\Delta t}{2}\right)$$

$$\mathbf{k}_3 = \Delta t\, \mathbf{f}\left(\mathbf{x} + \frac{\mathbf{k}_2}{2}, t + \frac{\Delta t}{2}\right)$$

$$\mathbf{k}_4 = \Delta t\, \mathbf{f}\left(\mathbf{x} + \mathbf{k}_3, t + \Delta t\right)$$

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \frac{\mathbf{k}_1}{6} + \frac{\mathbf{k}_2}{3} + \frac{\mathbf{k}_3}{3} + \frac{\mathbf{k}_4}{6} + O\left(\Delta t^5\right)$$

Numerical Integration of ODEs

- adaptive stepsize control
  - change step size according to the error
  - decrease/increase step size depending on whether the local error is high/low
  - higher integration speed in “simple” regions
  - good error control
- approaches:
  - stepsize doubling
  - embedded Runge-Kutta schemes

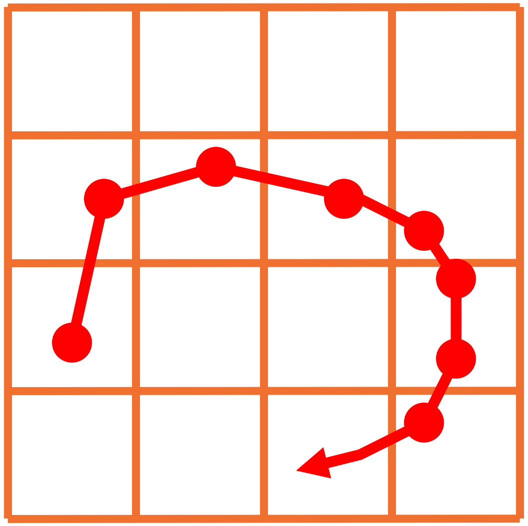
- further reading:
  - WH Press, SA Teukolsky, WT Vetterling, BP Flannery: *Numerical Recipes*

Particle Tracing on Grids

- vector field given on a grid
- solve

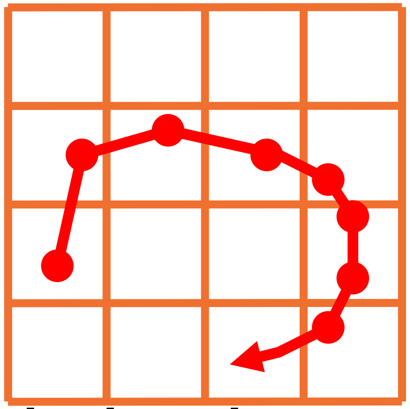
$$\mathbf{L}(0) = \mathbf{x}_0 \quad , \quad \frac{d\mathbf{L}(t)}{dt} = \mathbf{v}(\mathbf{L}(t), t)$$

  for the path line
- incremental integration
- discretized path of the particle



Particle Tracing on Grids

- most simple case: Cartesian grid



- basic algorithm:

```
Select start point (seed point)
Find cell that contains start point
While (particle in domain) do
  Interpolate vector field at
  current position
  Integrate to new position
  Find new cell
  Draw line segment between latest
  particle positions
Endwhile
```

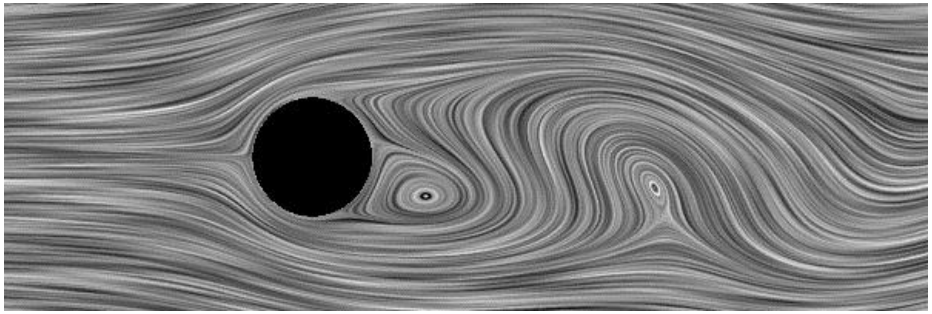
point location ***interpolation*** ***integration*** ***point location***

Line Integral Convolution

- a global method to visualize vector fields

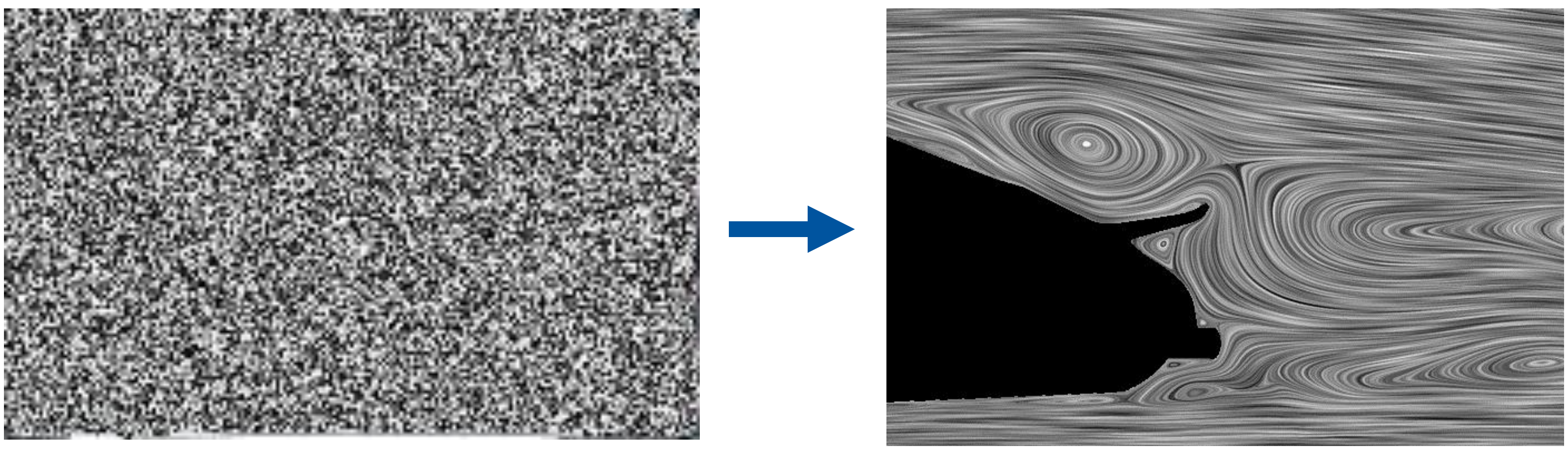
Line Integral Convolution

- line Integral Convolution (LIC)
  - visualize dense flow fields by imaging its integral curves
  - vover domain with a random texture (so called ‚input texture‘, usually stationary white noise)
  - blur (convolve) the input texture along the path lines using a specified filter kernel
- look of 2D LIC images
  - intensity distribution along path lines shows high correlation
  - no correlation between neighboring path lines



Line Integral Convolution

- idea of Line Integral Convolution (LIC)
  - global visualization technique
  - start with random texture
  - smear out along stream lines



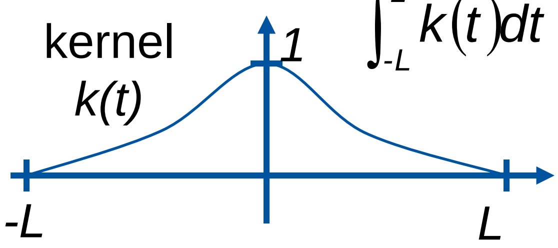
Line Integral Convolution

- algorithm for 2D LIC
  - let $t \rightarrow \Phi_0(t)$ be the path line containing the point (x_0,y_0)
  - $T(x,y)$ is the randomly generated input texture
  - compute the pixel intensity as:

$$I(x_0, y_0) = \int_{-L}^{L} k(t) \cdot T(\phi_0(t))\, dt$$

convolution with kernel

- kernel:
  - finite support $[-L,L]$
  - normalized
  - often simple box filter
  - often symmetric (isotropic)

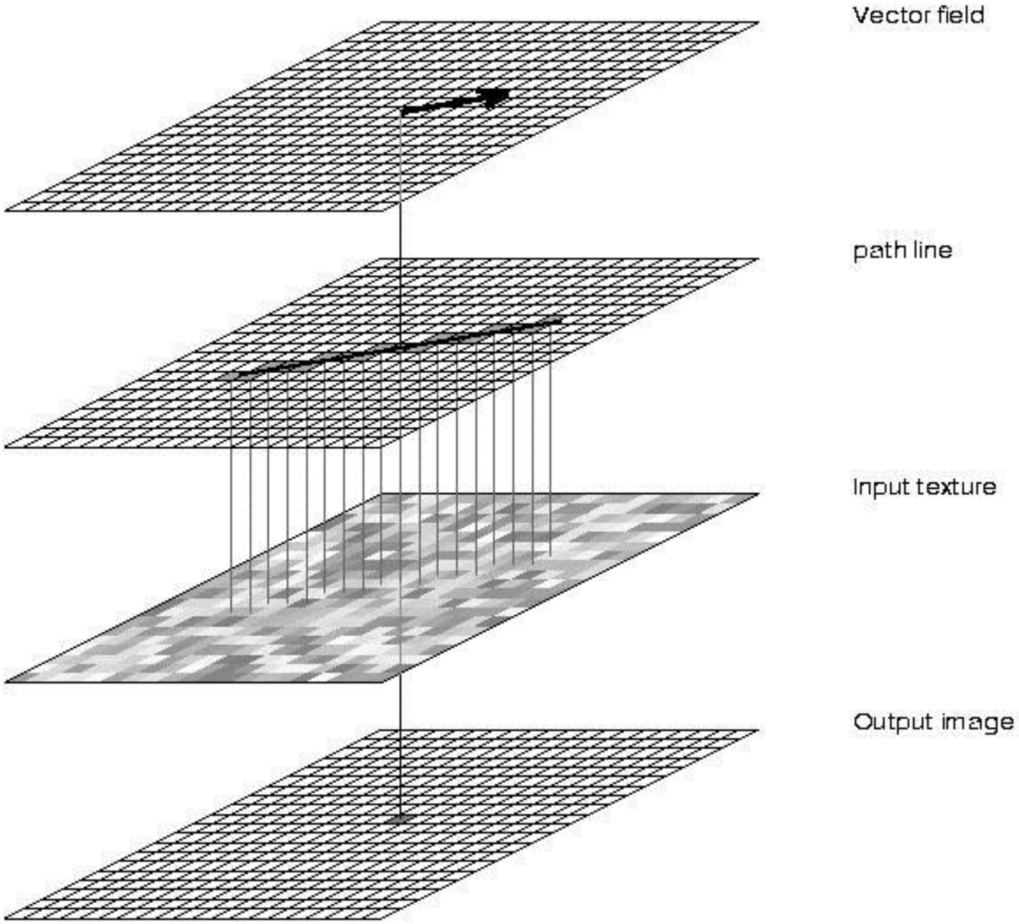


$$\int_{-L}^{L} k(t)\, dt = 1$$

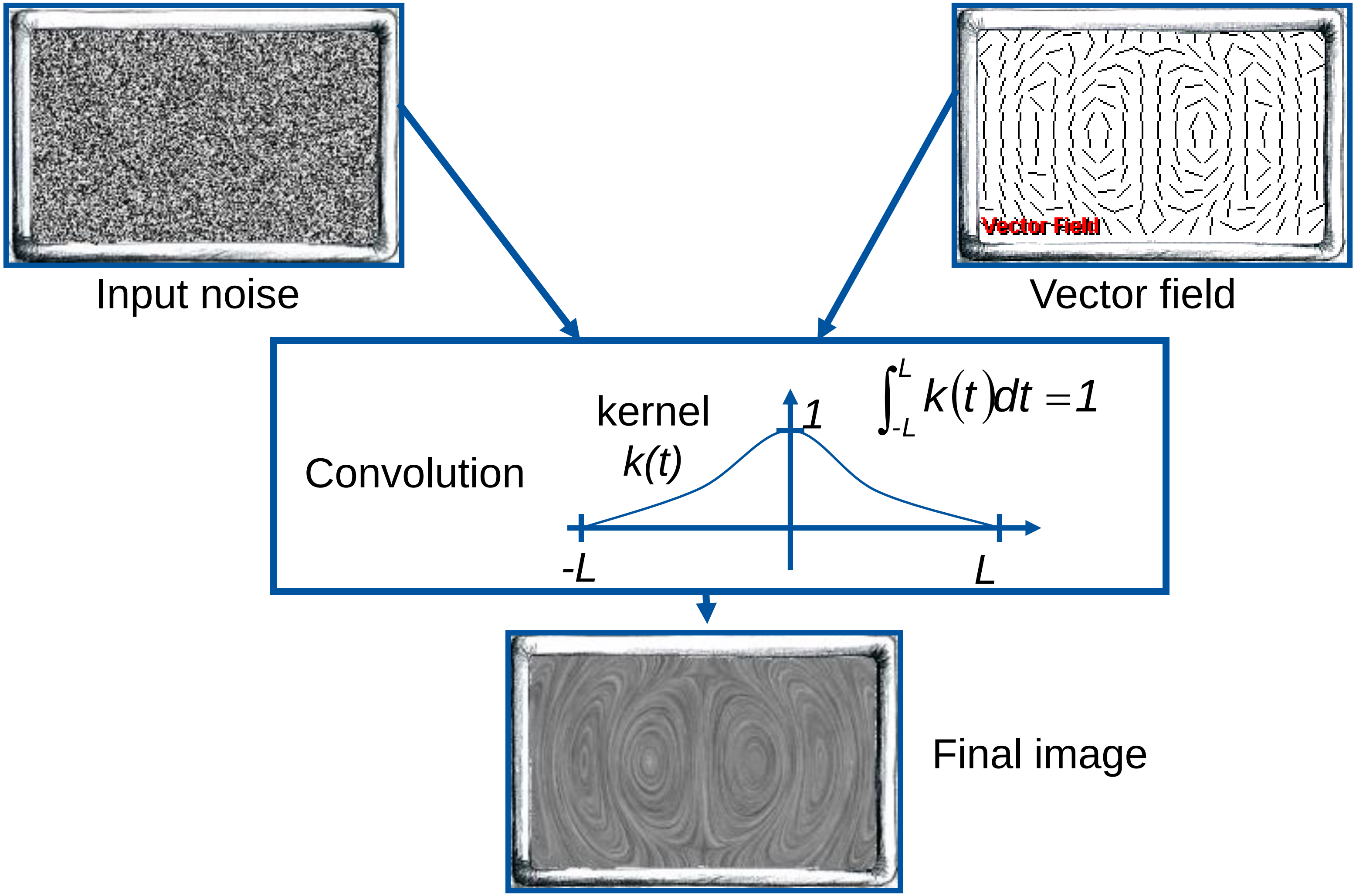
Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Line Integral Convolution

- algorithm for 2D LIC
  - convolve a random texture along the stream lines



Line Integral Convolution



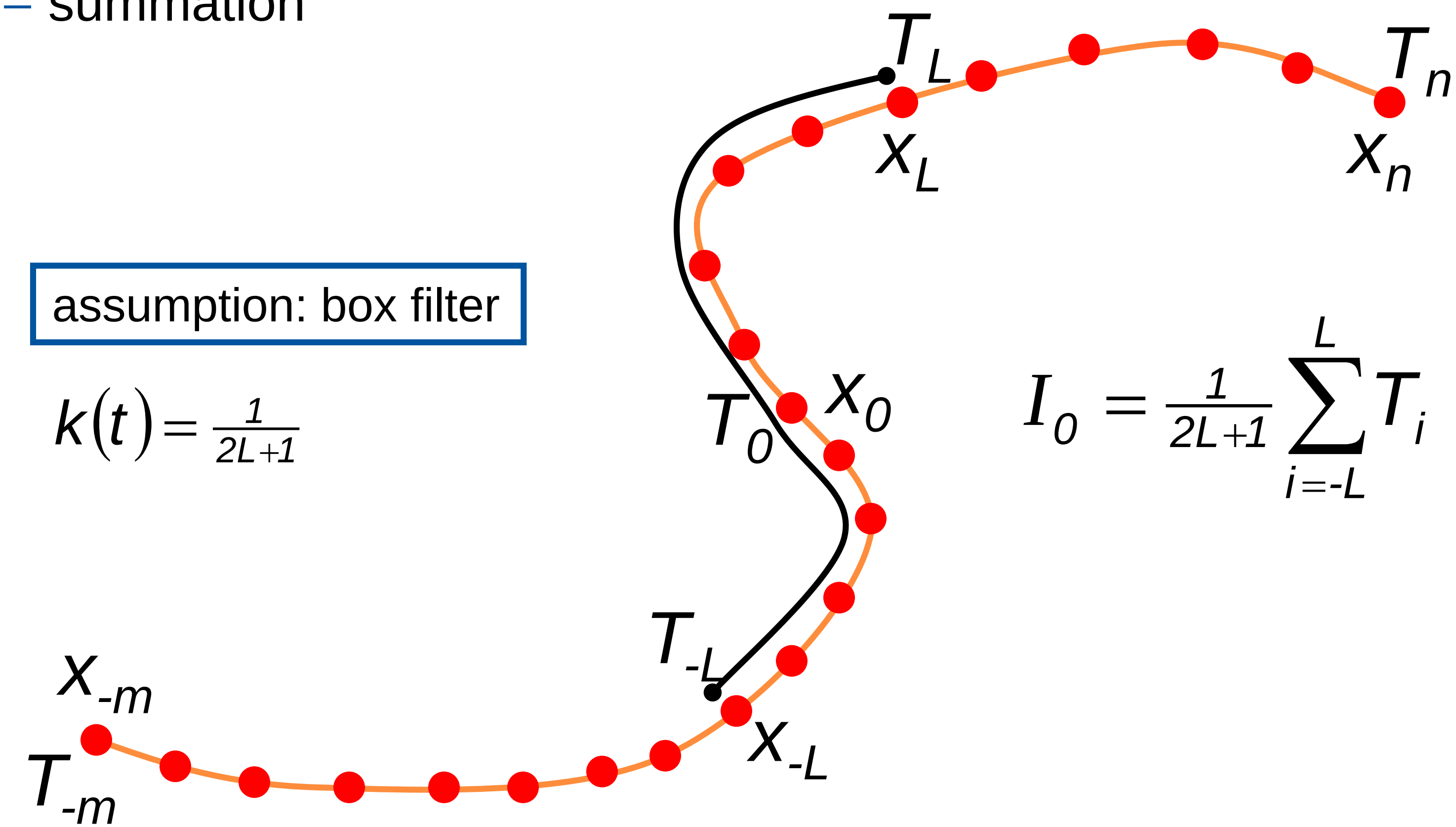
Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Line Integral Convolution

- fast LIC
- problems with LIC
  - new stream line is computed at each pixel
  - convolution (integral) is computed at each pixel
  - slow
- improvement:
  - compute very long stream lines
  - reuse these stream lines for many different pixels
  - incremental computation of the convolution integral

Line Integral Convolution

- fast LIC: incremental integration
  - discretization of convolution integral
  - summation



assumption: box filter

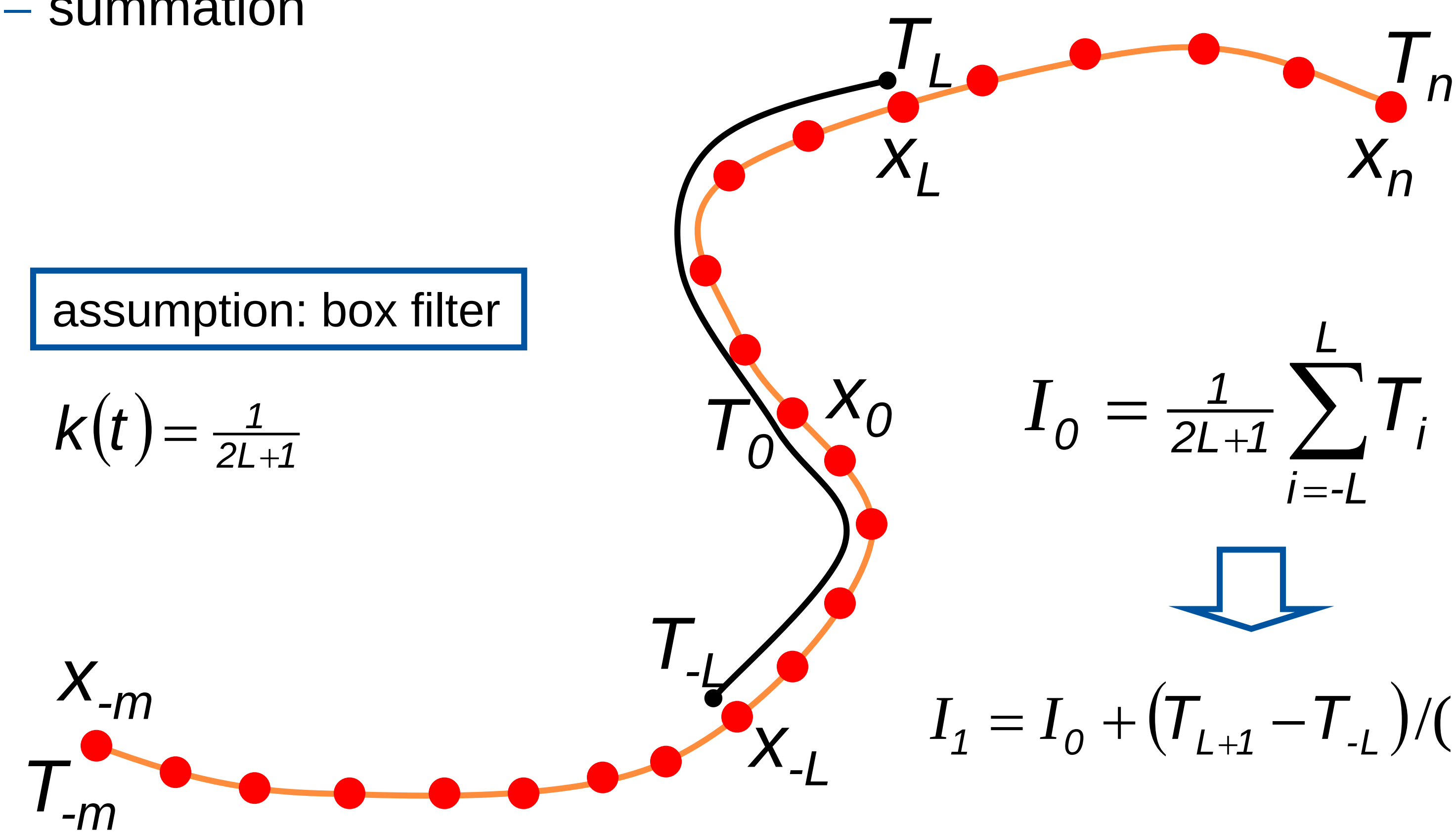
$$k(t) = \frac{1}{2L+1}$$

$$I_0 = \frac{1}{2L+1} \sum_{i=-L}^{L} T_i$$

Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

Line Integral Convolution

- fast LIC: incremental integration
  - discretization of convolution integral
  - summation



assumption: box filter

$$k(t) = \frac{1}{2L+1}$$

$$I_0 = \frac{1}{2L+1} \sum_{i=-L}^{L} T_i$$

$$I_1 = I_0 + \left(T_{L+1} - T_{-L}\right)/(2L+1)$$

Line Integral Convolution

- fast LIC: incremental integration for constant kernel
  - stream line $\mathbf{x}_{-m},\ldots,\mathbf{x}_0,\ldots,\mathbf{x}_n$ with $m,n \geq L$
  - given texture values $T_{-m},\ldots,T_0,\ldots,T_n$
  - what are results of convolution: $I_{-m+L},\ldots,I_0,\ldots,I_{n-L}$?
  - for box filter (constant kernel):

$$I_0 = \tfrac{1}{2L+1}\sum_{i=-L}^{L} T_i$$

  - incremental integration:

$$I_{j+1} - I_j = \frac{1}{2L+1}\sum_{i=-L}^{L}\left(T_{i+j+1} - T_{i+j}\right) = \frac{1}{2L+1}\left(T_{L+j+1} - T_{-L+j}\right)$$

Line Integral Convolution

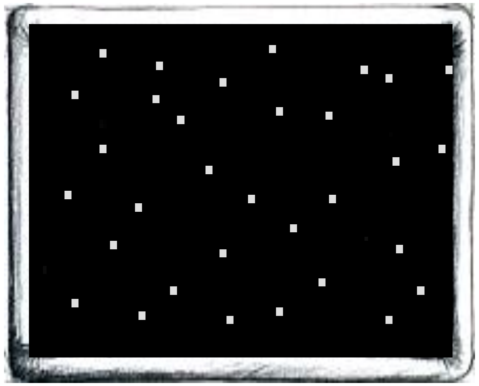
- fast LIC: Algorithm
  - data structure for output: Luminance/Alpha image
    - luminance = gray-scale output
    - alpha = number of streamline passing through that pixel

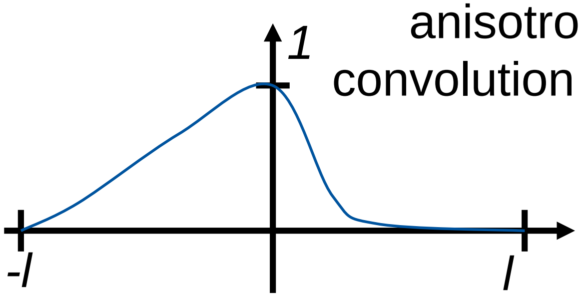
```
For each pixel p in output image
    If (Alpha(p) < #min) Then
        Initialize streamline computation with x0 = center of p
        Compute convolution I(x0)
        Add result to pixel p
        For m = 1 to Limit M
            Incremental convolution for I(xm) and I(x-m)
            Add results to pixels containing xm and x-m
        End for
    End if
End for
Normalize all pixels according to Alpha
```


Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

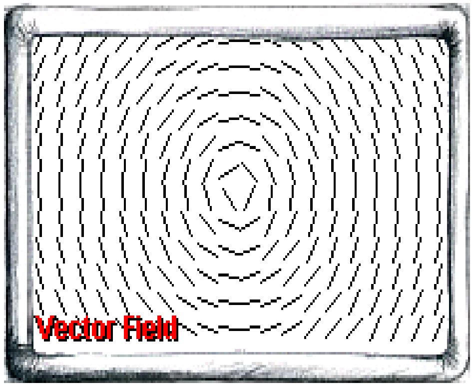
Line Integral Convolution

- oriented LIC (OLIC):
  - visualizes orientation (in addition to direction)
  - sparse texture
  - anisotropic convolution kernel
  - acceleration: integrate individual drops and compose them to final image



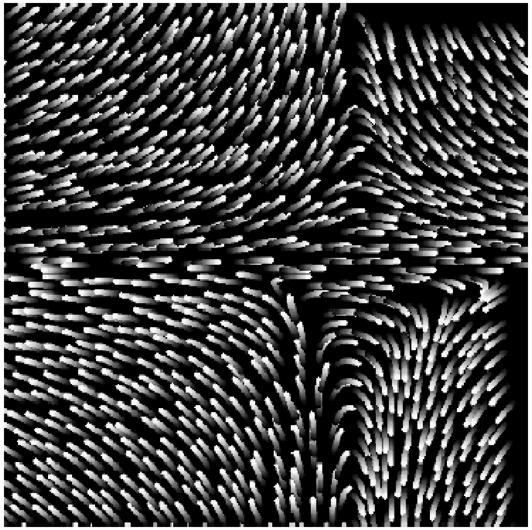


anisotropic convolution kernel



Line Integral Convolution

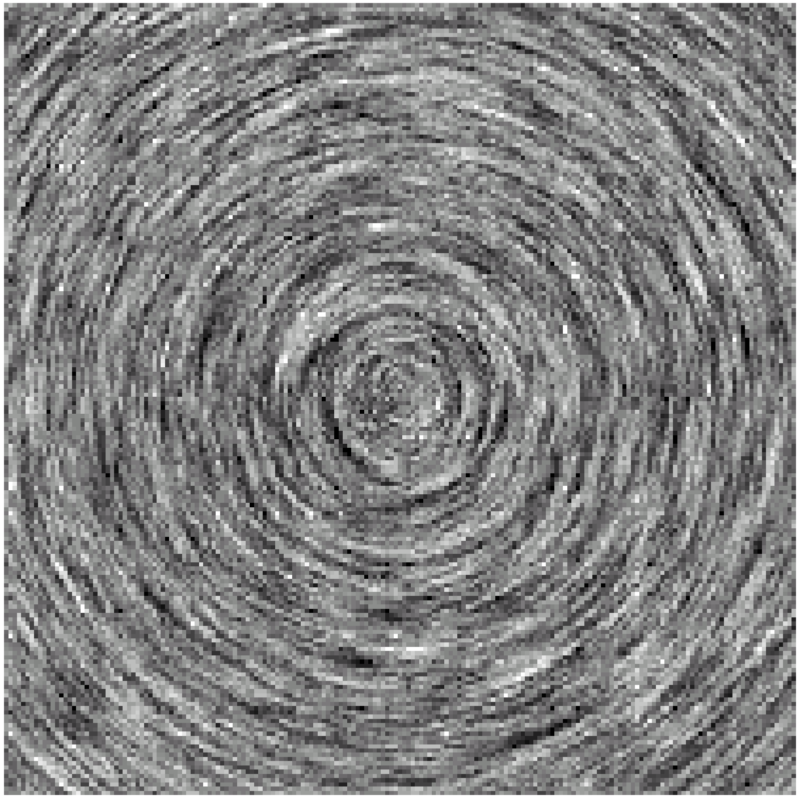
- oriented LIC (OLIC)

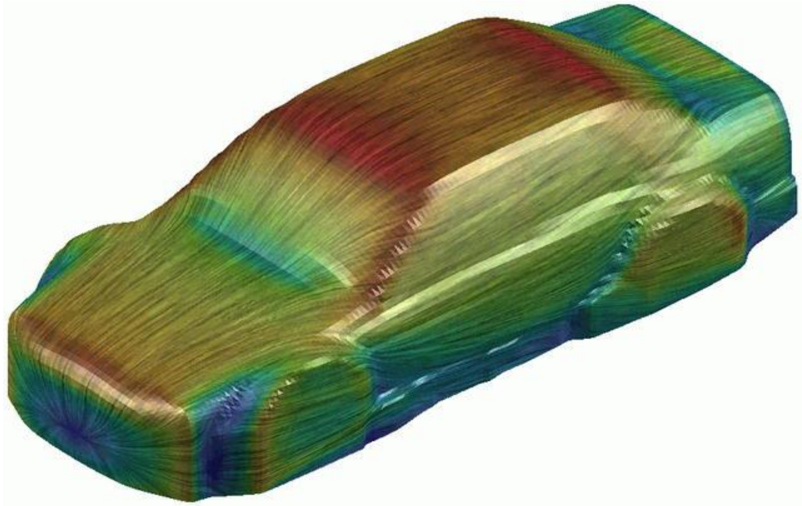


Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

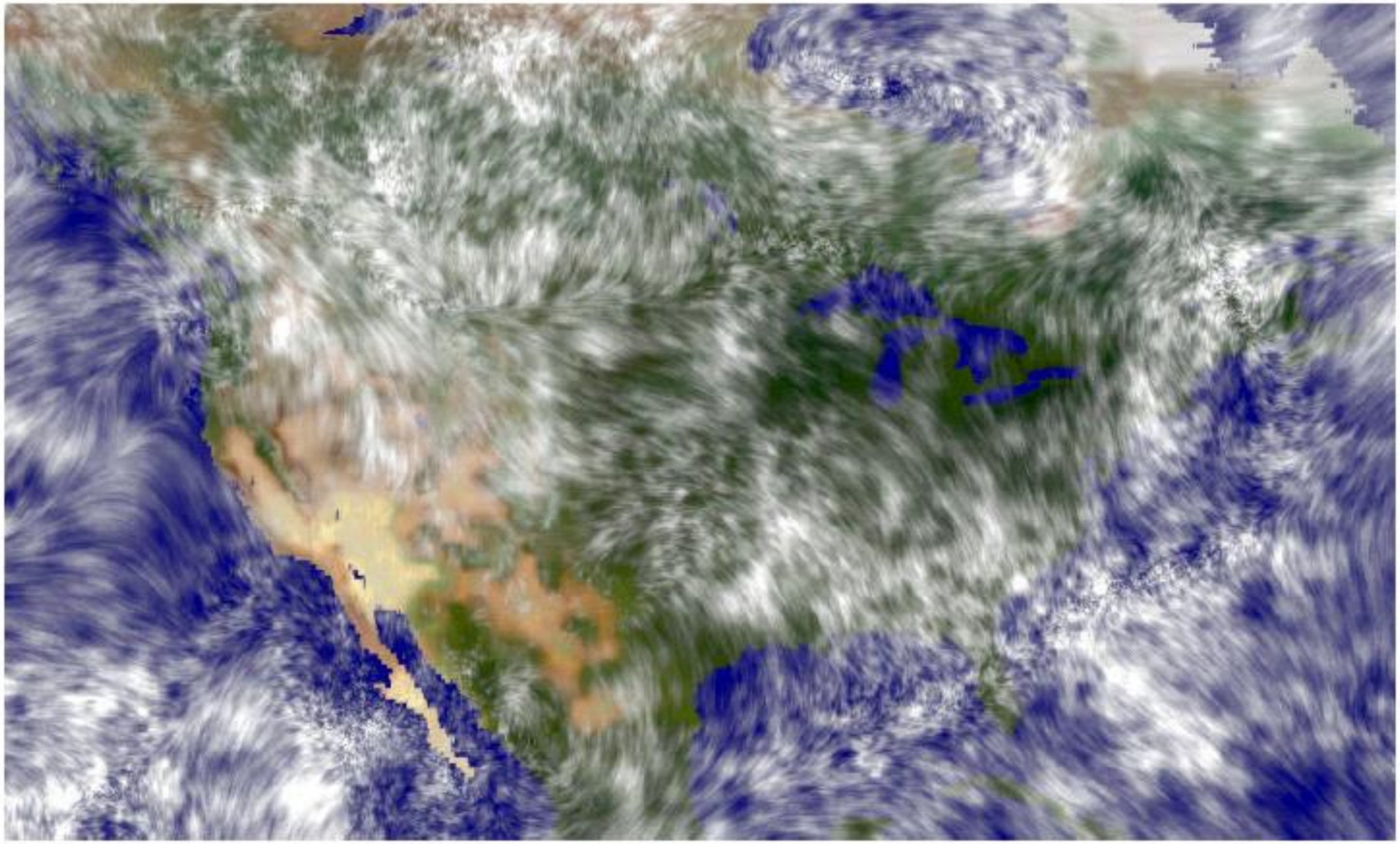
Line Integral Convolution

- LIC - Line Integral Convolution



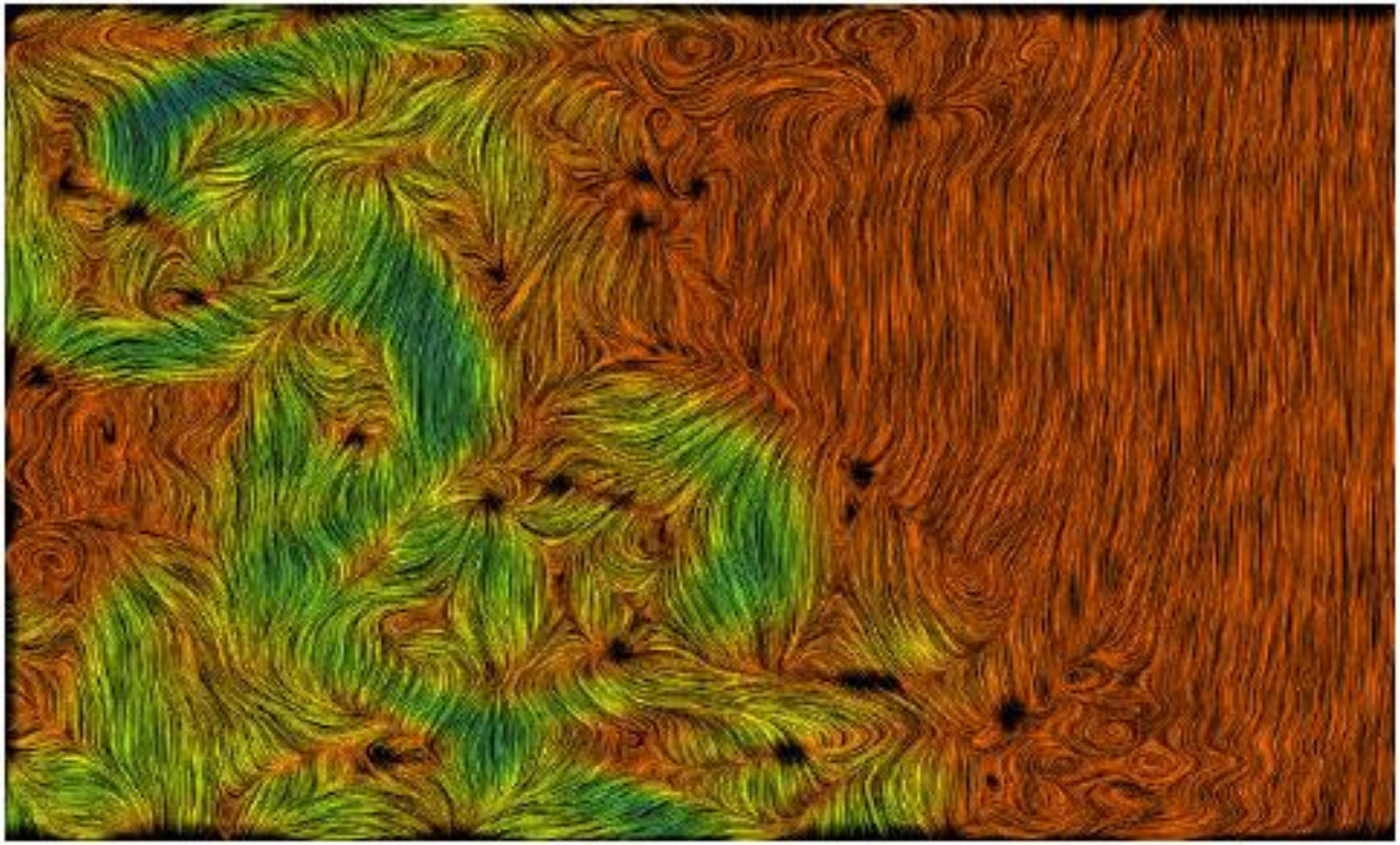


Line Integral Convolution



Lic-Applications: length of convolution integral with respect to magnitude of vector field

Line Integral Convolution



Lic and color coding of velocity magnitude

Line Integral Convolution

- summary:
  - dense representation of flow fields
  - convolution along stream lines $\rightarrow$ correlation along stream lines
  - for 2D and 3D flows
  - stationary flows
  - extensions:
    - Unsteady flows
    - Animation
    - Texture advection

3D Vector Fields

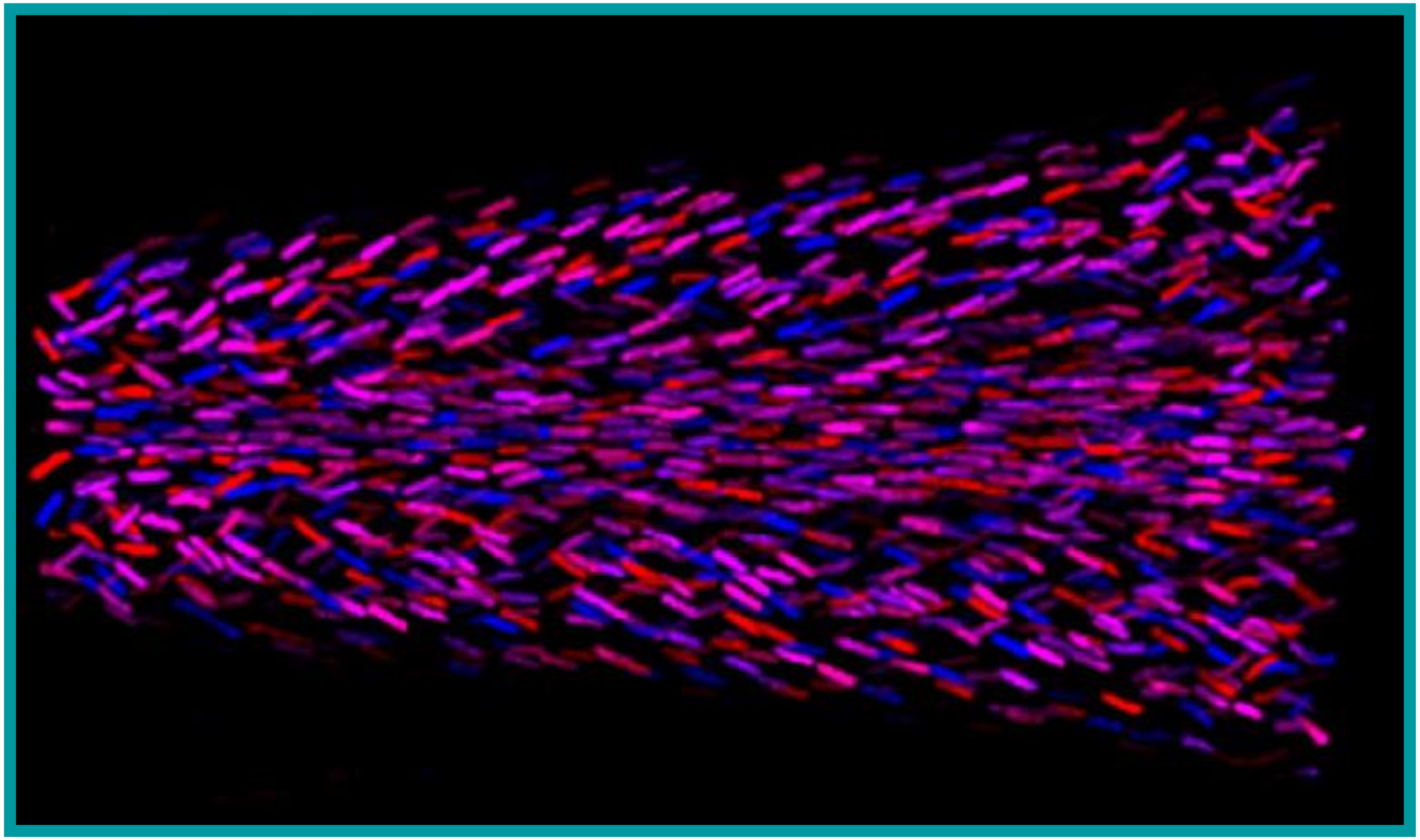
- most algorithms can be applied to 2D and 3D vector fields
- main problem in 3D: effective mapping to graphical primitives
- main aspects:
  - occlusion
  - amount of (visual) data
  - depth perception

3D Vector Fields

- approaches to occlusion issue:
  - sparse representations
  - animation
  - color differences to distinguish separate objects
  - continuity
- reduction of visual data:
  - sparse representations
  - clipping
  - importance of semi-transparency

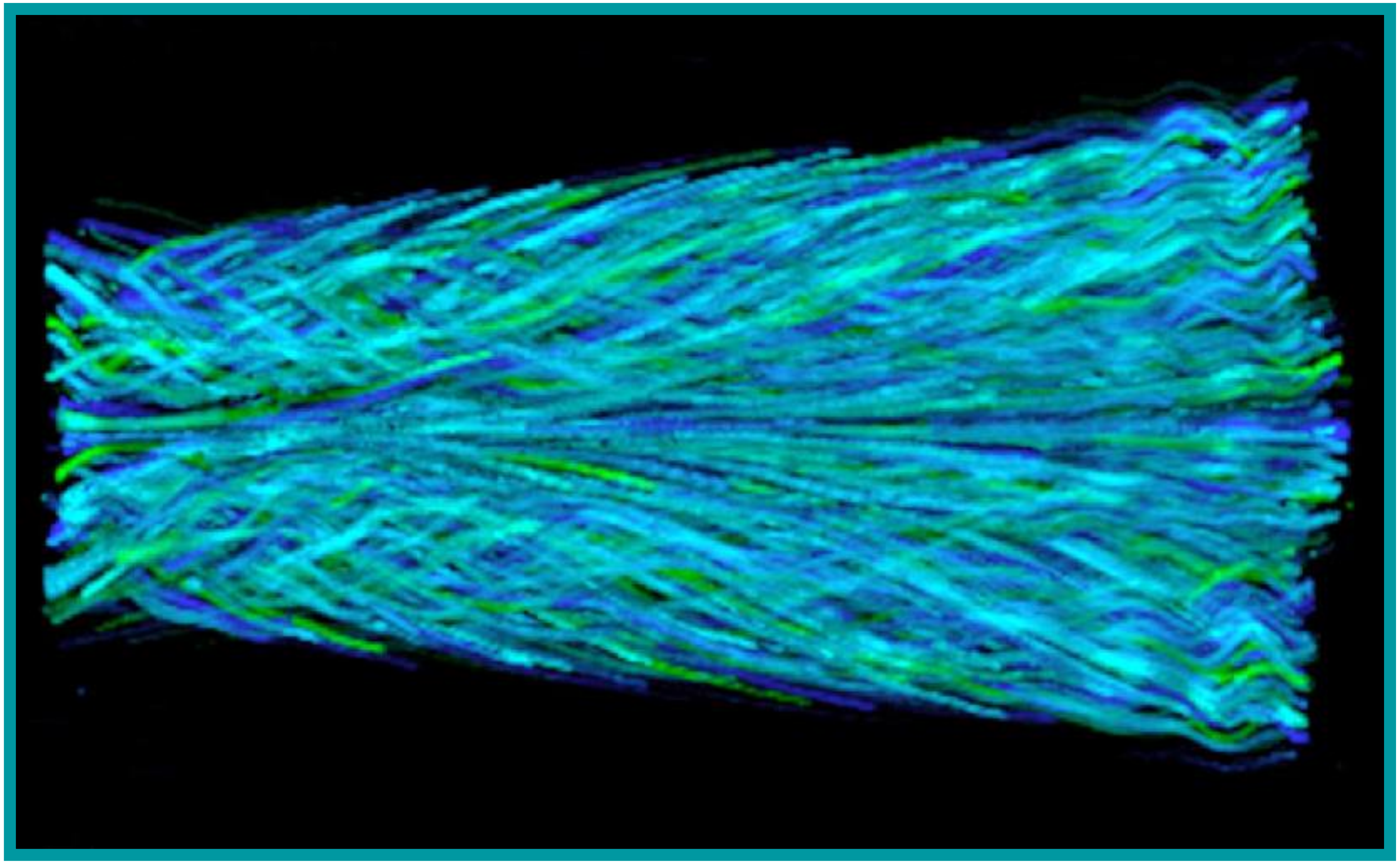
3D Vector Fields

- missing continuity



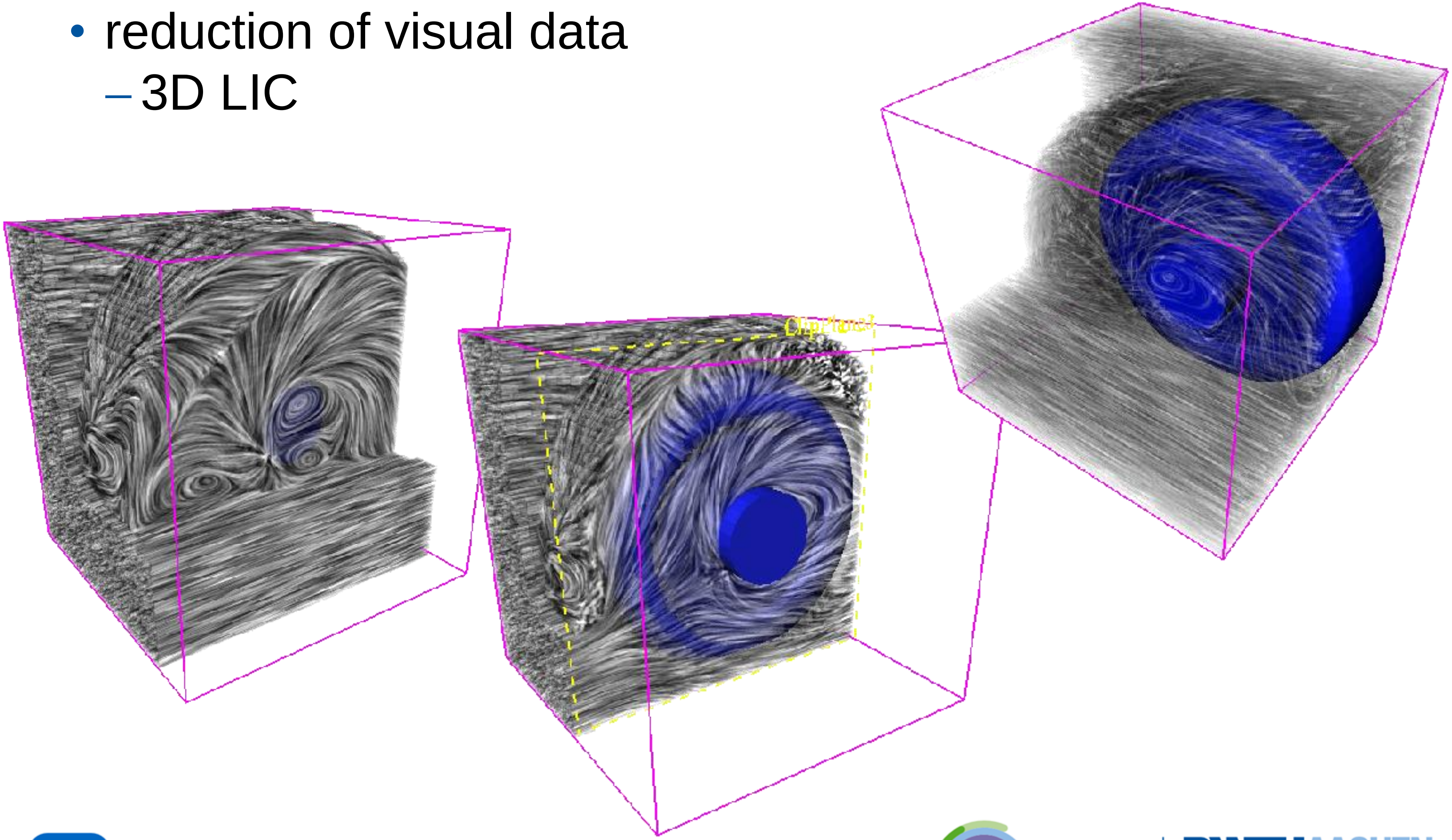
3D Vector Fields

- color differences to identify connected structures



3D Vector Fields

- reduction of visual data
  - 3D LIC

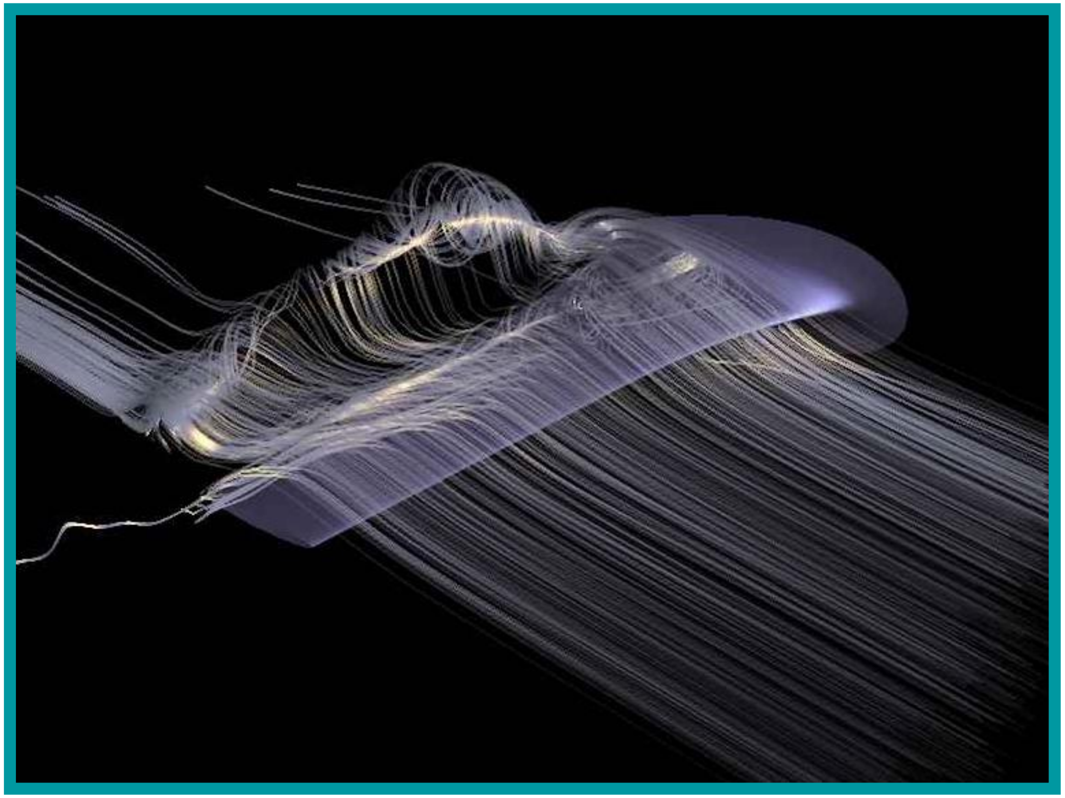


3D Vector Fields

- improving spatial perception:
  - depth cues
    - perspective
    - occlusion
    - motion parallax
    - stereo disparity
    - color (atmospheric, fogging)
  - halos
  - orientation of structures by shading (highlights)

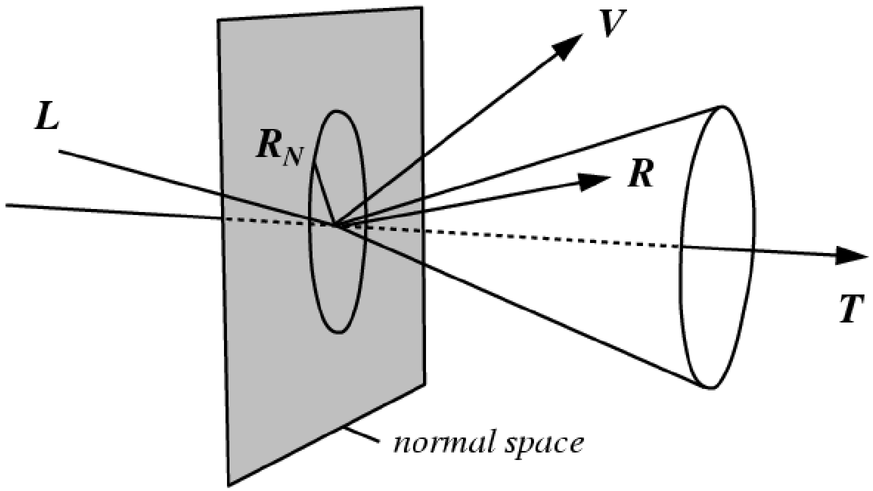
3D Vector Fields

- illumination



3D Vector Fields

- illuminated streamlines [Zöckler et al. 1996]
  - model: streamline is made of thin cylinders
  - problem
    - no distinct normal vector on surface
    - normal vector in plane perpendicular to tangent: normal space
    - cone of reflection vectors



Visual Computing Institute | Prof. Dr. Leif Kobbelt
Computer Graphics and Multimedia
Geometry Processing

3D Vector Fields

- illuminated streamlines (*cont.*)
  - light vector is split in tangential and normal parts

$$\begin{aligned}\mathbf{V}\cdot\mathbf{R} &= \mathbf{V}\cdot\left(\mathbf{L}_T-\mathbf{L}_N\right)=\mathbf{V}\cdot\left((\mathbf{L}\cdot\mathbf{T})\mathbf{T}-(\mathbf{L}\cdot\mathbf{N})\mathbf{N}\right)\\ &= (\mathbf{L}\cdot\mathbf{T})(\mathbf{V}\cdot\mathbf{T})-(\mathbf{L}\cdot\mathbf{N})(\mathbf{V}\cdot\mathbf{N})\\ &= (\mathbf{L}\cdot\mathbf{T})(\mathbf{V}\cdot\mathbf{T})-\sqrt{1-(\mathbf{L}\cdot\mathbf{T})^2}\sqrt{1-(\mathbf{V}\cdot\mathbf{T})^2}\\ &= f\left((\mathbf{L}\cdot\mathbf{T}),(\mathbf{V}\cdot\mathbf{T})\right)\end{aligned}$$

  - Idea: Represent *f*() by 2D texture
  - Access pre-computed f() during rendering